System i Developer

Create REST Web Services from RPG Programs (and SQL!)

Jon Paris

Jon.Paris @ Partner400.com
www.Partner400.com

Partner400

Notes

About Jon Paris (that's me):

My career in IT spans 50+ years including a 12 year period with IBM's Toronto Laboratory. One of my proudest achievements during that period was being a founding member of the small team that designed and built RPG IV.

I am the co-founder of Partner400, a firm specializing in customized education and mentoring services for IBM i (AS/400, System i, iSeries, etc.) developers. I am also a founding member of the System i Developer education consortium which, among other things organized the twice yearly RPG & DB2 Summit.

Together with my partner Susan Gantner, I devote my time to educating developers on techniques and technologies to extend and modernize their applications and development environments. We also write frequently for IT Jungle's RPG Guru column (www.itjungle.com). We have also written many times for IBM Systems Magazine and other publications. You can find our "collected works" at our Authory site https://authory.com/jonparisandsusangantner

Feel free to contact me any time: Jon.Paris @ partner400.com

Creating REST Web Services - Basics

REST = Representational State Transfer

- Doesn't really explain a lot though does it !

Basically a URL that identifies a processing resource

- Think of it as being a request for a "web page" designed to be filled in or consumed by a program

The HTTP request type differentiates the required actions

- GET retrieves information
  - This is the one we will use - but IWS supports others too
- PUT is used to create a new item
- POST will completely update an item (or create it if it does not exist)
- There's also PATCH (Partial Update) and DELETE

Input and output data can be in any format

- For REST services input data is frequently part of the URL &/or the query string
  - If in the body it is most commonly coded as JSON or XML

IBM supply us with the Integrated Web Services Server (IWS)

- This will be the tool I will be looking at today
- It provides an easy-to-use option for quickly deploying web services
  - Using regular programs or SQL statements for the processing

IBM's IWS - ibm.com/systems/i/software/iws/

Available for several years now

- I first wrote about it back in 2012

Originally we recommended clients use it only for Proof of Concept

- i.e. to demonstrate that IBM i could actually play in the web service arena

Over time it has improved significantly

- And can now be a good candidate for production workloads

Turns a simple *PGM, *SRVPGM or SQL statement into a Web Service

- The IBM supplied wizard does all the work

IWS can create SOAP or REST services

- And is frequently enhanced by IBM

For details go to:

- ibm.com/support/pages/integrated-web-services-ibm-i-web-services-made-easy
  - Or simply Google IWS Home Page site:IBM.com
- You'll find the documentation links at the foot of the page

But before you can start using it ...

IWS Pre-Requisites and Documentation

Partner400

These products are require - But you'll almost certainly have them installed already

- Qshell - base option 30 of operating system
- PASE - base option 33 of operating system
- Host Servers - base option 12 of operating system
- Digital Certificate Manager - base option 34 of operating system
- IBM HTTP Server for IBM i

Optionally:

- IBM Java SE 7 32 bit must be loaded for Qshell script support
  - IBM recommend loading Java SE 8 64 bit
    - That ensures that any servers that are created use the latest supported runtime.

Make sure all current group PTFs are applied

- Particularly the HTTP Server PTFs
  - New features are usually added to IWS via PTFs

Documentation

- Integrated Web Services Server - Administration and Programming Guide
  - 2018 version on web - not completely up-to-date
    - Links are on the IWS main page as noted on previous page
  - For generating IWS services using SQL see:
    - developer.ibm.com/tutorials/creating-rest-apis-based-on-sql-statements/

Getting Started ...

Begin by connecting to the Admin server:

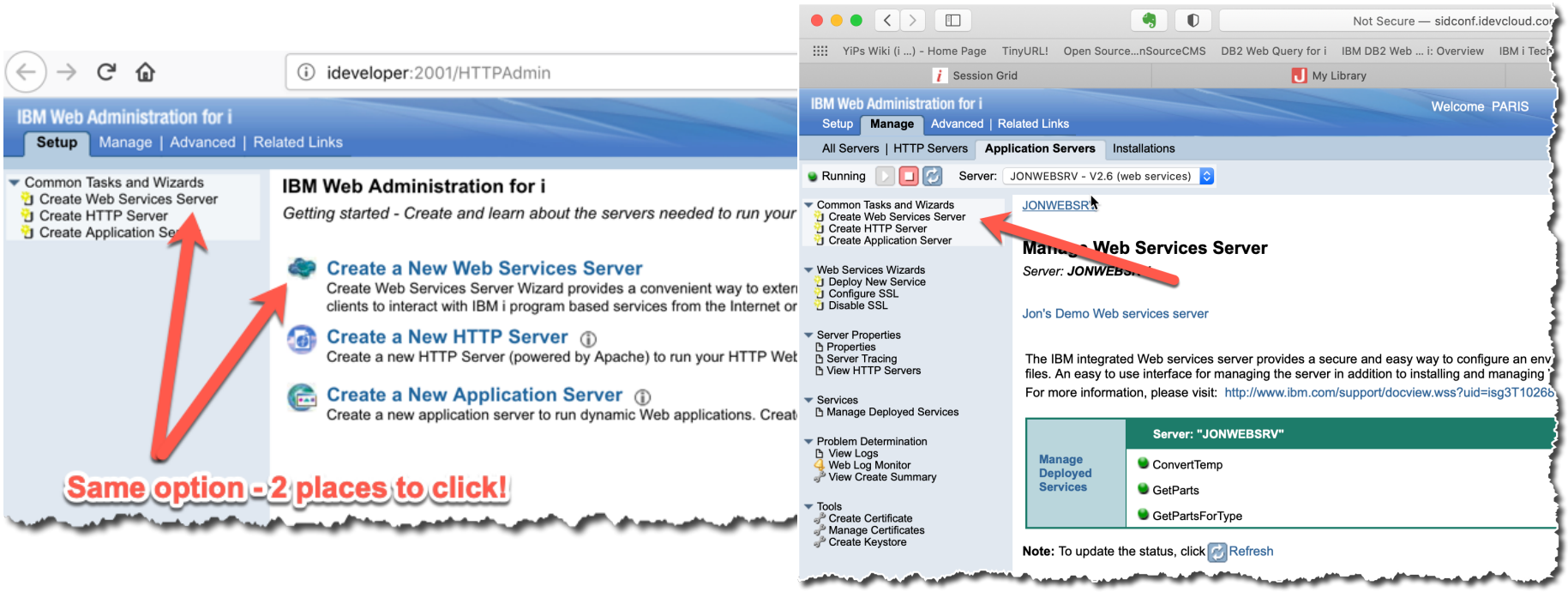
- http://YourServerName:2001/HTTPAdmin

You will be asked to sign in and one of the screens below will eventually appear:

- If you have previously defined web servers you will see the screen on the right
  - Otherwise you should see the one on the left

Click on "Create ... Web Services Server"

- Note that there is no difference between the links shown
- They will all take you to the same place



Name Your Server

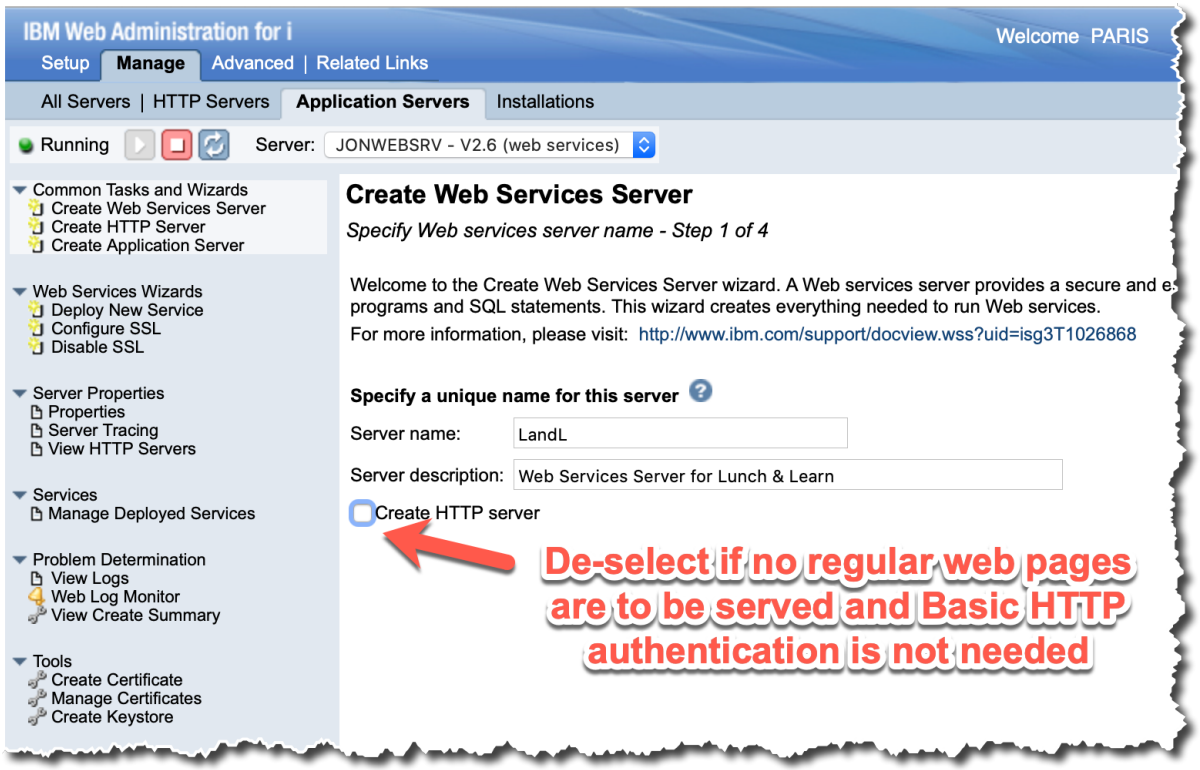
Partner400

Every server needs a name

- This can be any name of your choosing
  - But make it meaningful

Enter a name and then press **"Next"**

- **You can also enter a description if you wish**
  - **Probably a good idea for a production server**



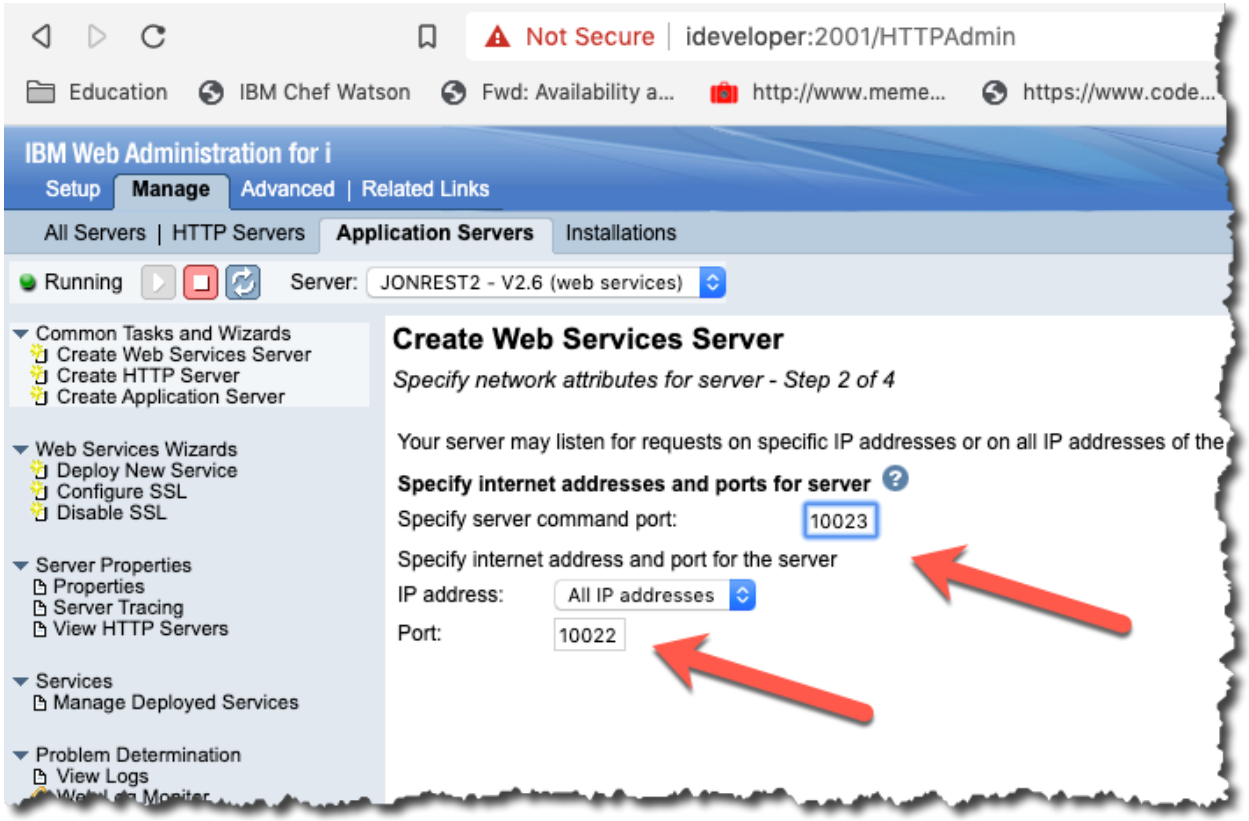
Identify the Ports to use

Each server uses two ports

- One for managing the server (10023 in this example)
- And a second which is the actual service point (10022)

IWS automatically selects ports for you

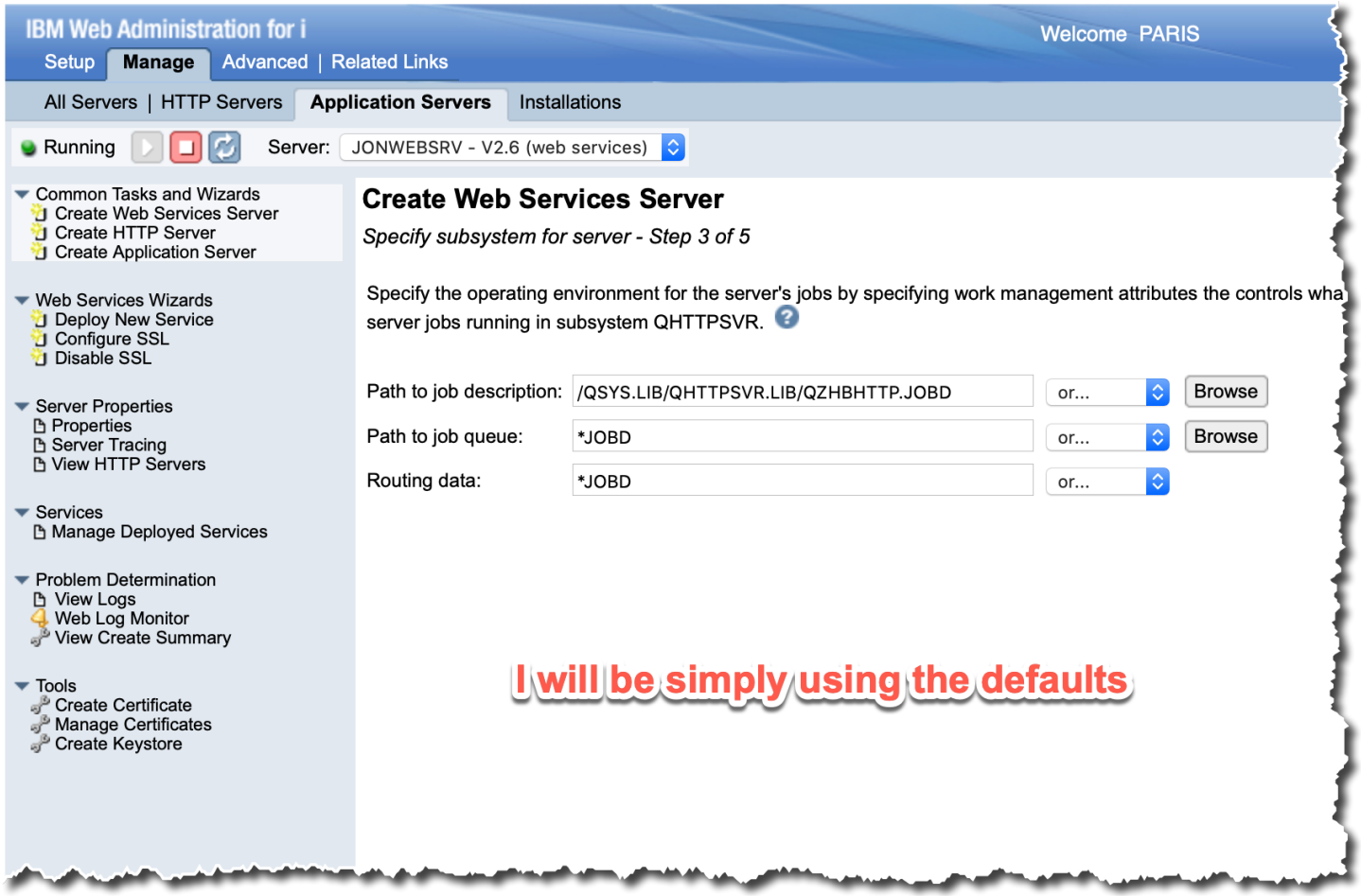
- But you can change them if needed
  - e.g. To match open ports on your firewall



Server Operating Environment

Releases 7.3+ allow you to control the server subsystem etc.

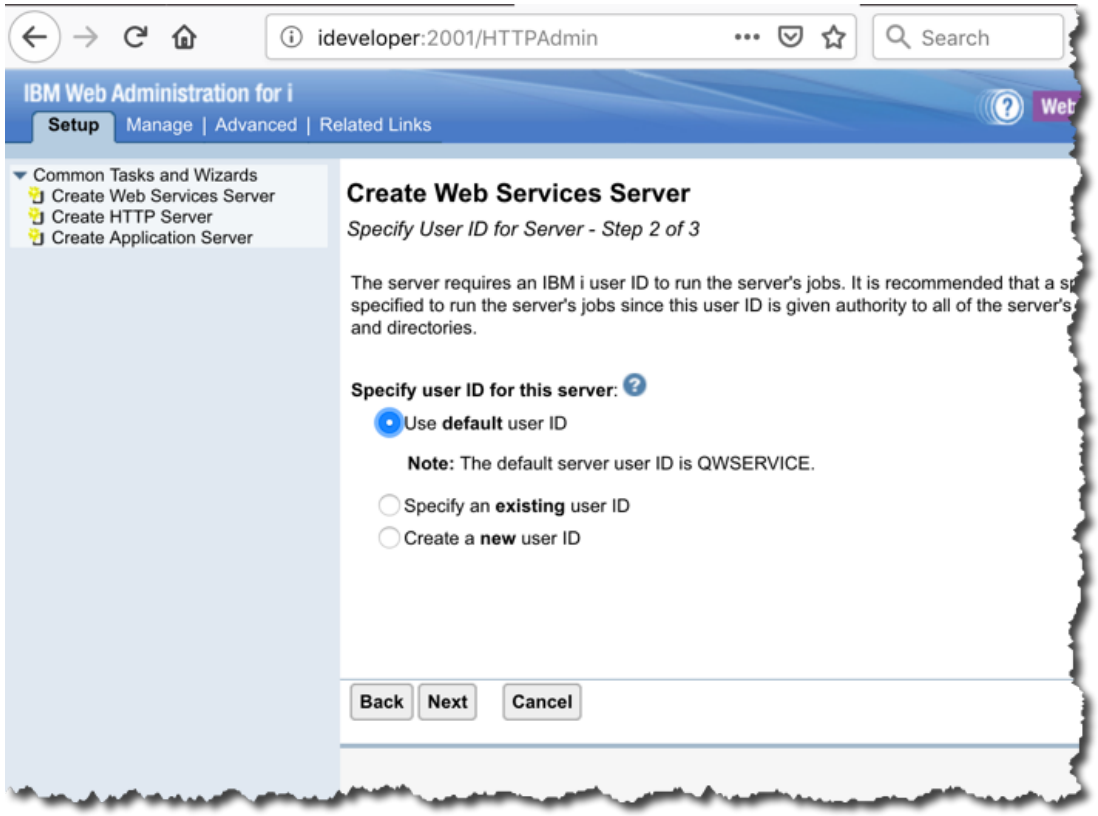
- If you take the defaults it will run in QHTTPSVR
  - Earlier releases don't show this screen and will default to QHTTPSVR



Select the User Profile

For proof of concept purposes the default user works fine

- For production it is best to create a new user that only has the needed authorities
- The manual describes the minimum authorities needed
  - Make sure the user is also authorized to any libraries needed by your service



First Steps - Completing the Task

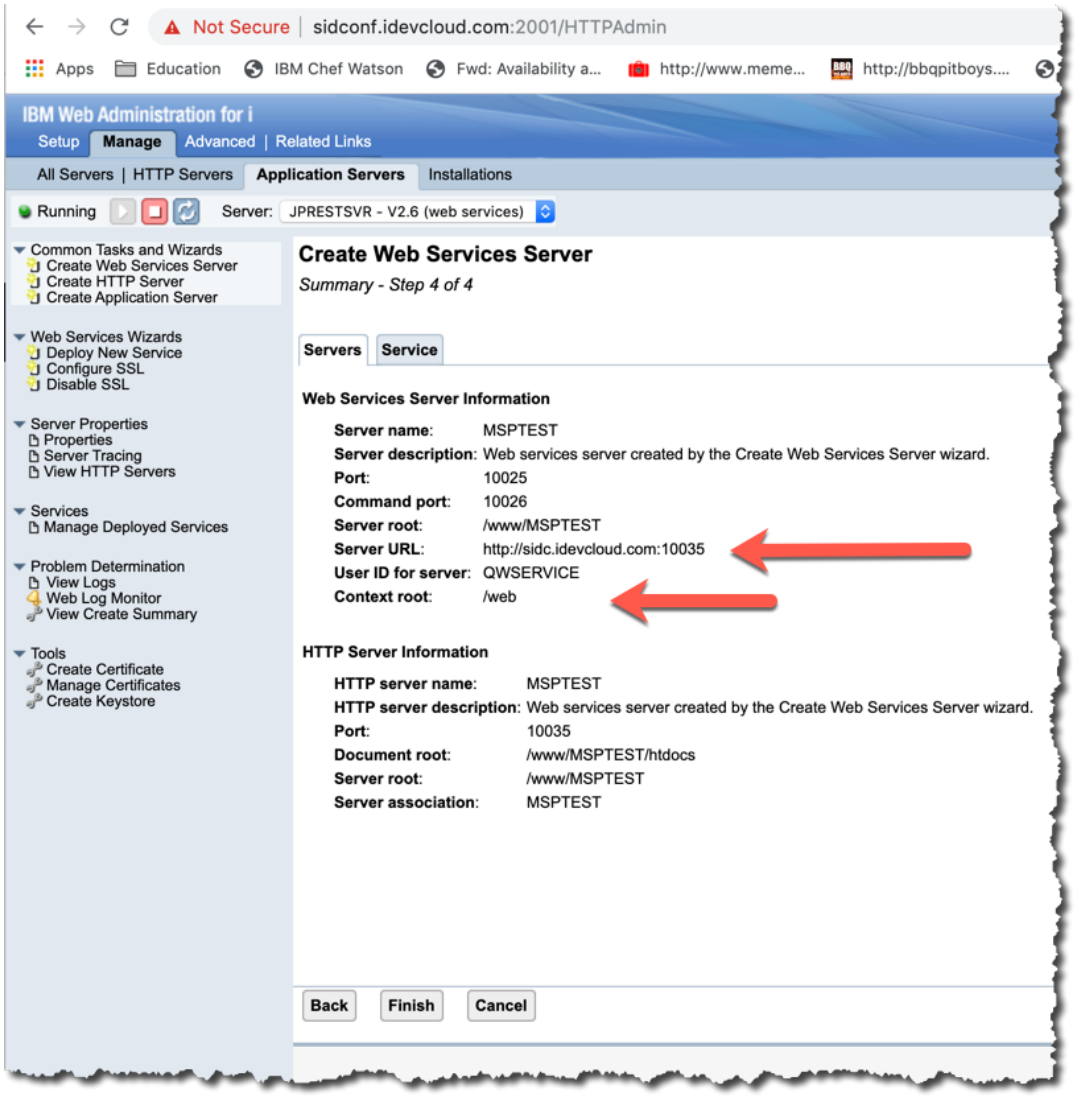
Partner400

This page summarizes what the wizard is building for you

- Note the Server URL
- And the context root

Press "Finish"

- This will actually build the server
- And start it running



First Steps - Progress Display

Partner400

This page appears during the build process

- You normally have to press the Refresh button to see the server page
  - I have never been able to work out when it does it automagically and when you have to manually request a refresh

First Steps - The Finished Server

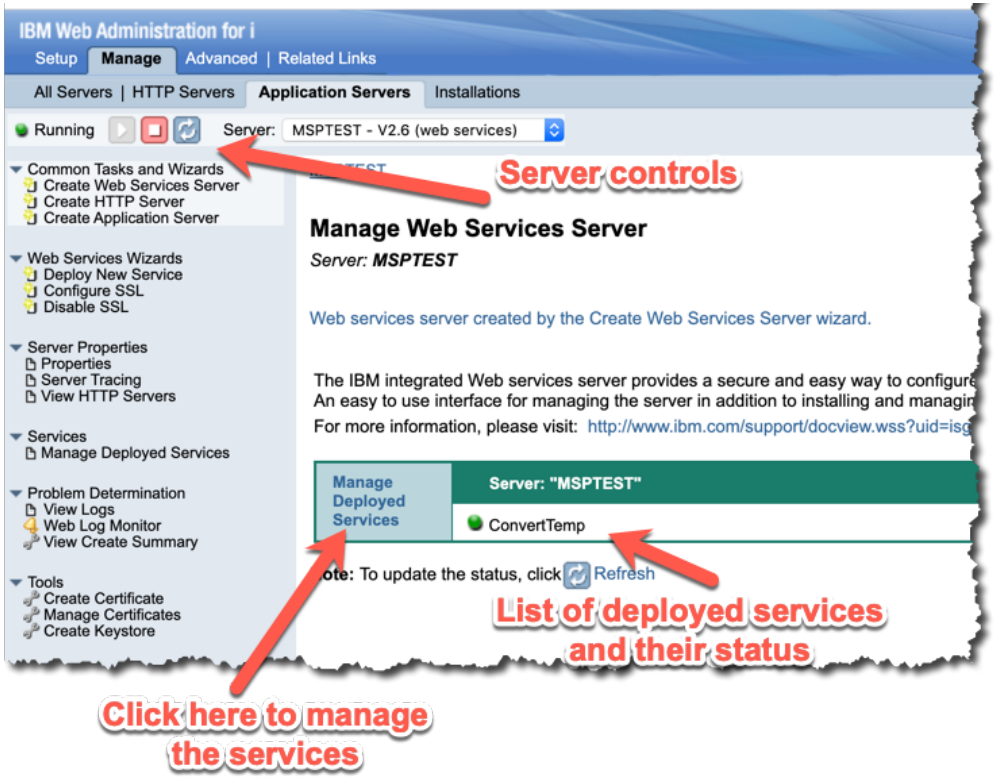
Partner400

A sample SOAP service has been deployed for you

- If you wish you can use it to test the basic server operation
  - I normally just delete it

To deploy a new service we need to "Manage ..." the services

- So click the marked big green box

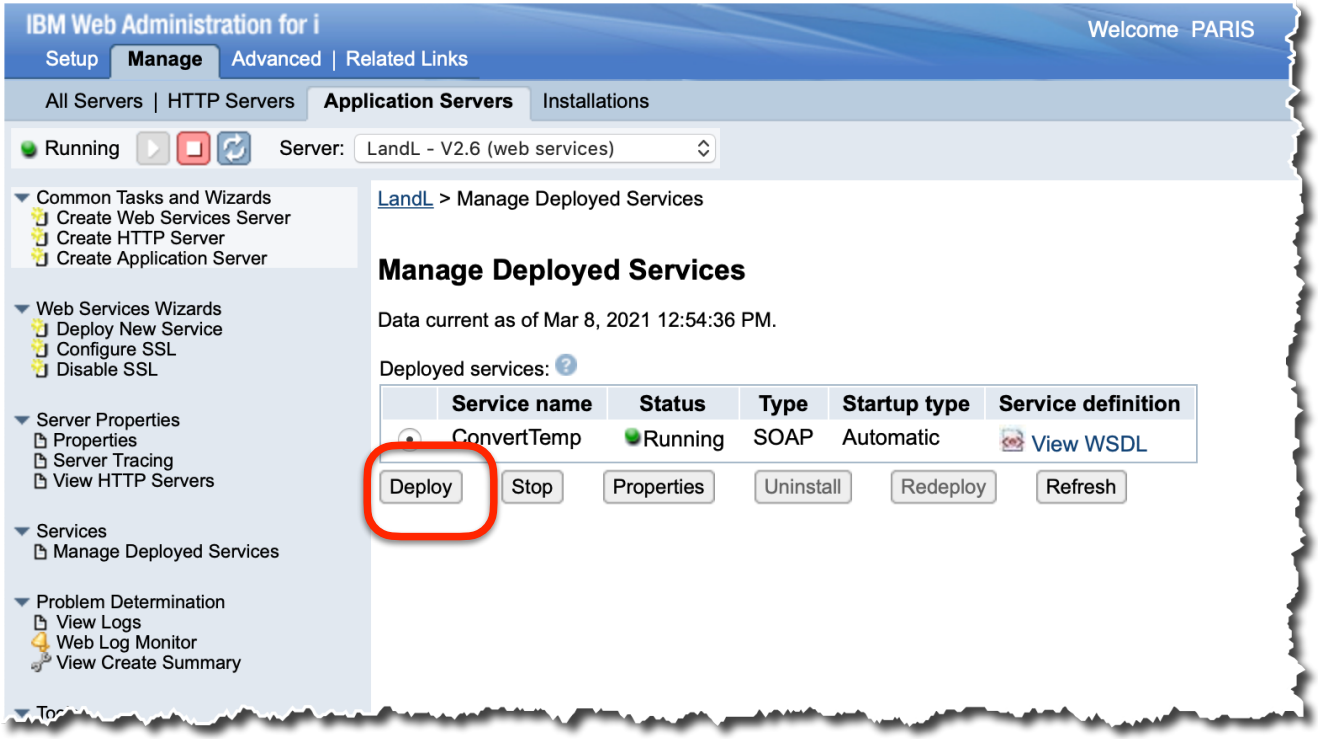


Service Management

Use this to:

- Create new services (Deploy)
  - And to Modify (Redeploy), Stop, Start, and Uninstall them
- You can also review a service's properties

We'll just create (Deploy) a new one



Deploying SQL as a Web Service

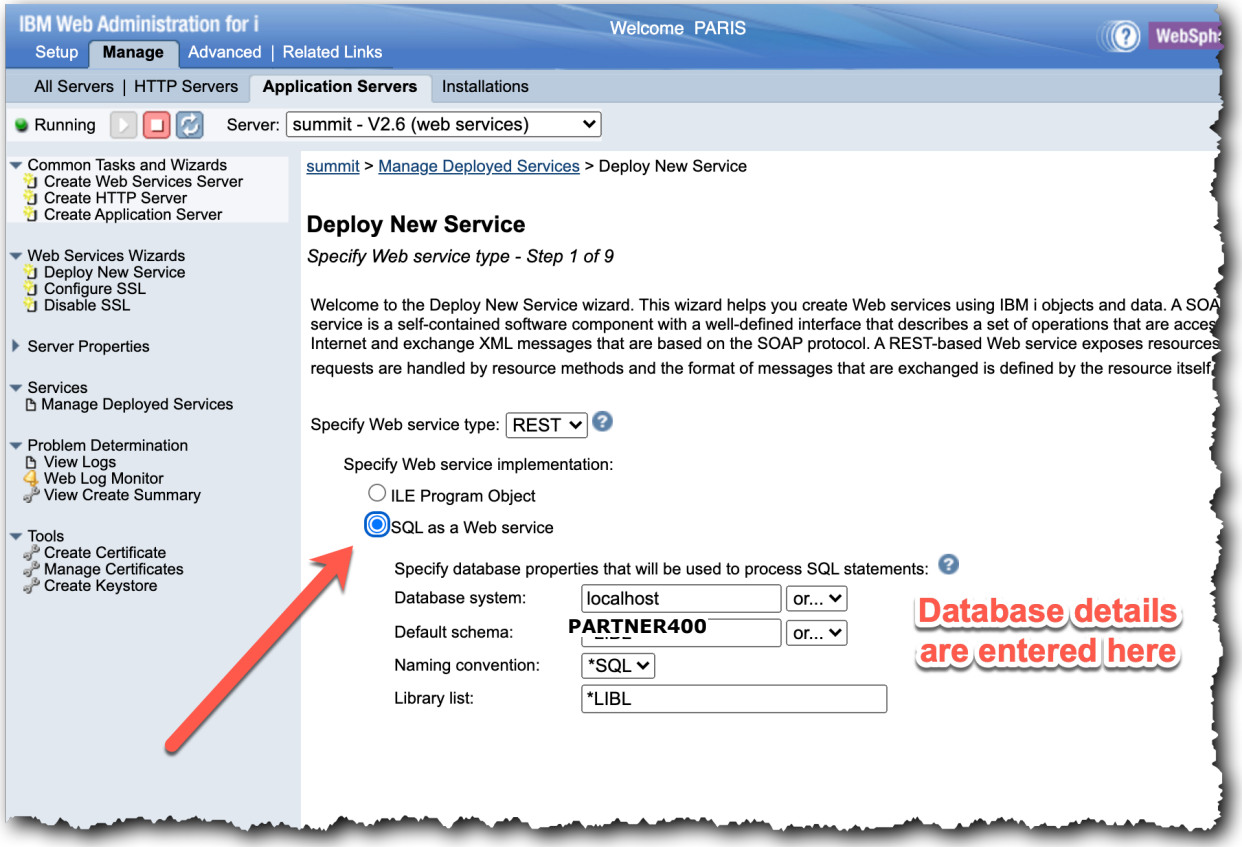
Paris Partner400

Your SQL skills determine how complex a response can be built

- You will see how the SQL is coded in a moment

You can specify a library (schema) to use as the default

- That way you don't always have to fully qualify the table names



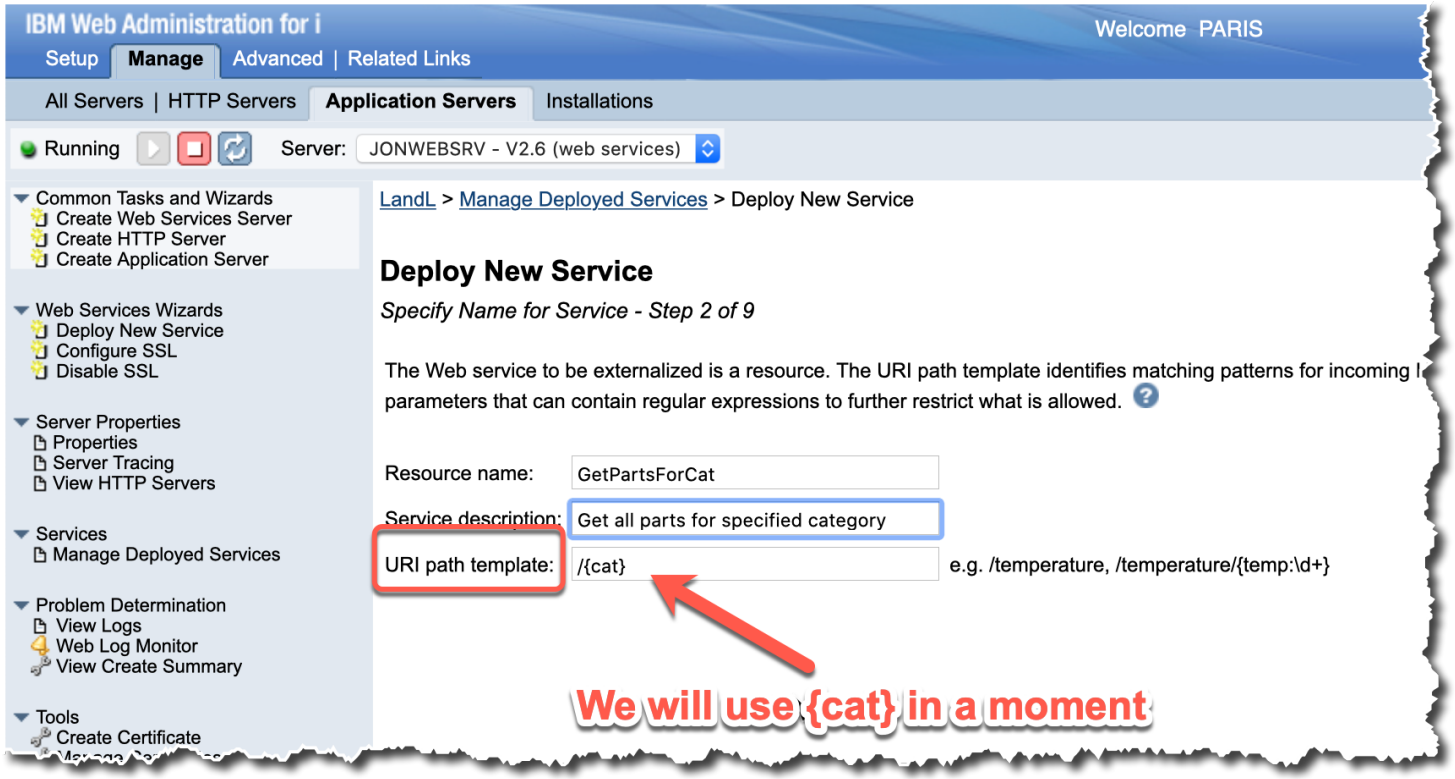
Deploying an SQL Service (Cont)

Here we get to supply the name for the service

- GetPartsForCat

And specify how the requested parameter will be passed

- In this case in the URL immediately following the last /
- You will see how to associate the name {cat} with the data in a moment



Deploying an SQL Service (Cont)

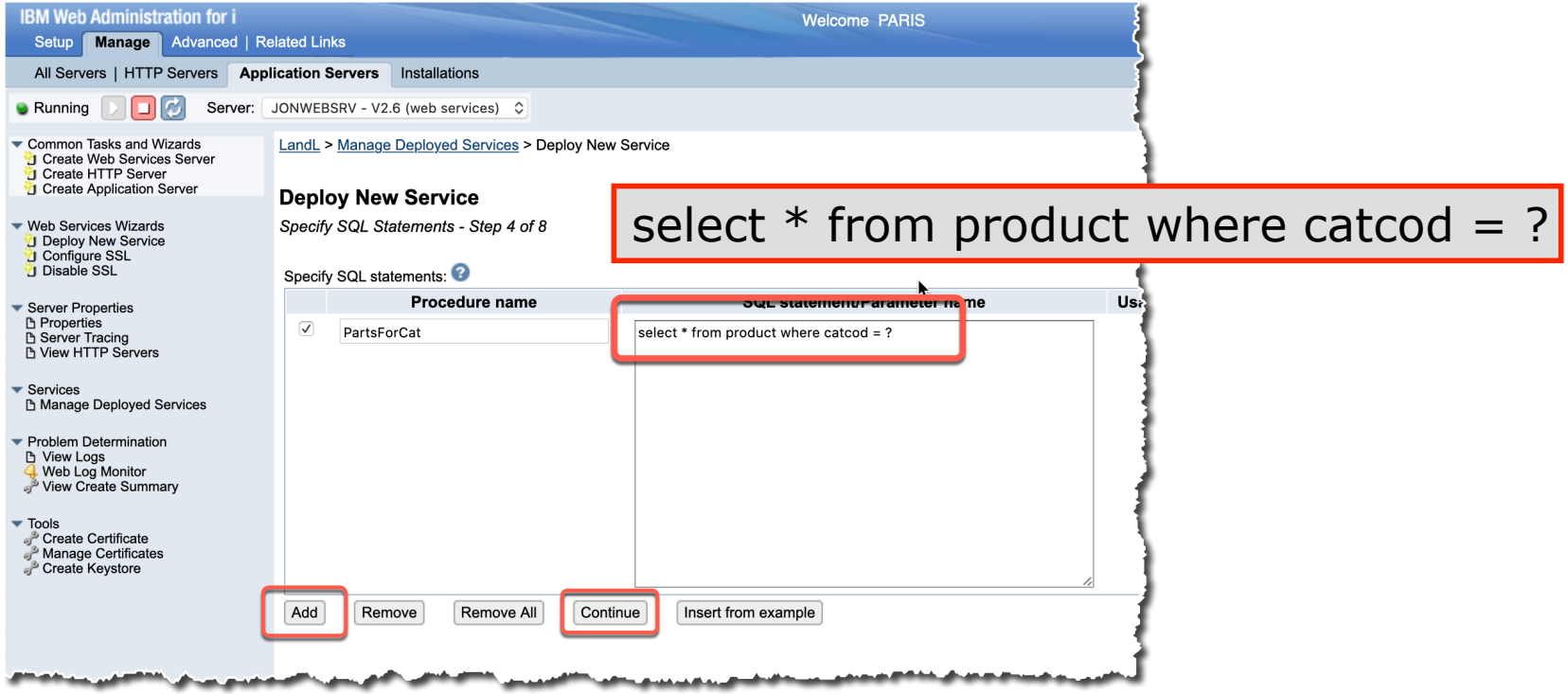
Time to enter the SQL statement(s)

Enter the SQL using the standard "?" as a parameter marker

- The wizard makes up a name for the parameter
  - We will change it to a meaningful name next

Use the Continue button when all SQL has been entered

- or use the Add button if you need additional SQL statements



Deploying an SQL Service (Cont)

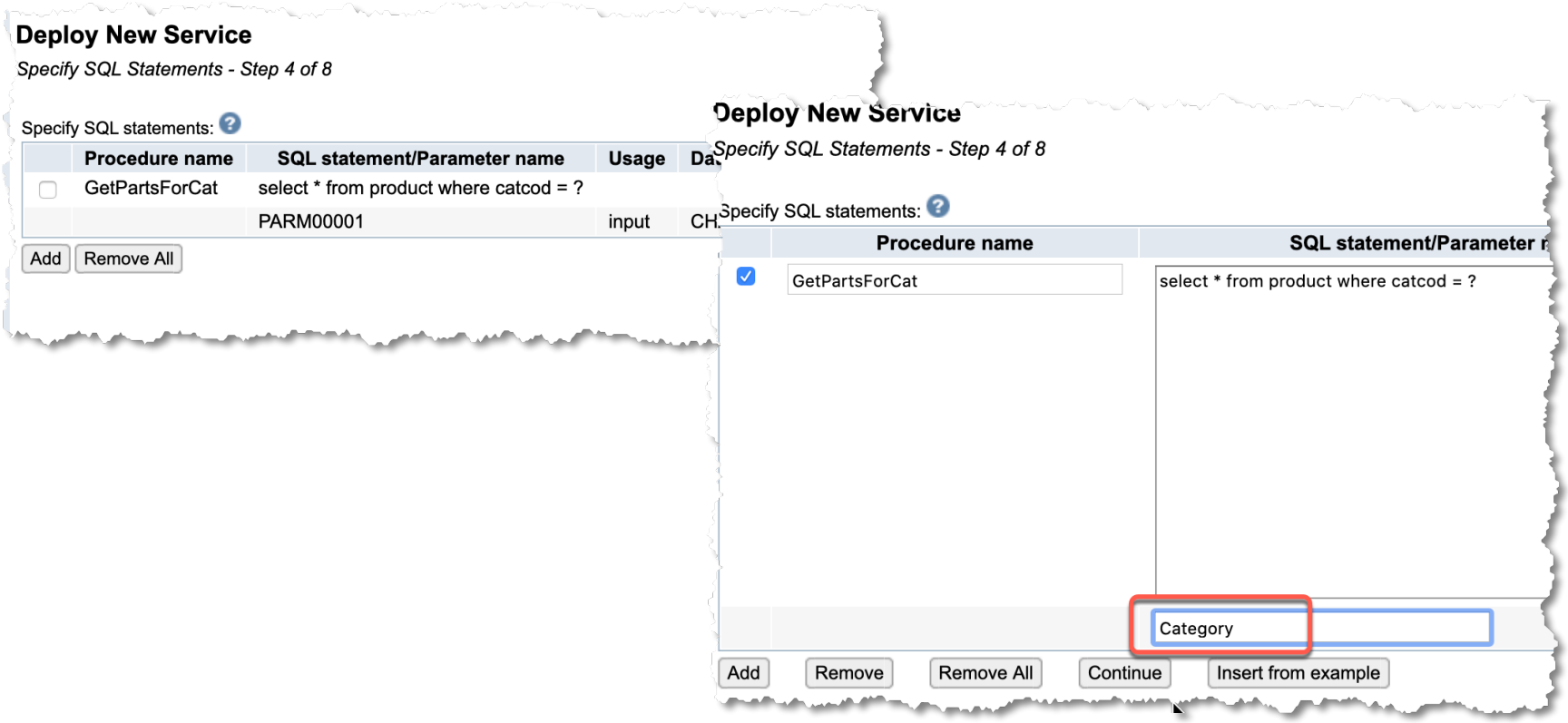
Partner400

Once the Continue button has been pressed this appears

- Notice the ugly name PARM00001

To change it select the procedure

- The second screen (which looks much like the original) will appear
  - But this time you can change the parameter name



Deploying an SQL Service (Cont)

Partner400

You can specify single row

- Or multi-row result

Trimming options on results

What constitutes an error

- Specify a custom error message

What HTTP status to return

Etc. etc.

Deploying an SQL Service (Cont)

Partner400

To assign the input parameter ...

- For the parameter Category select *PATH_PARAM as the source
- The identifier "cat" will appear

That's all we have to do

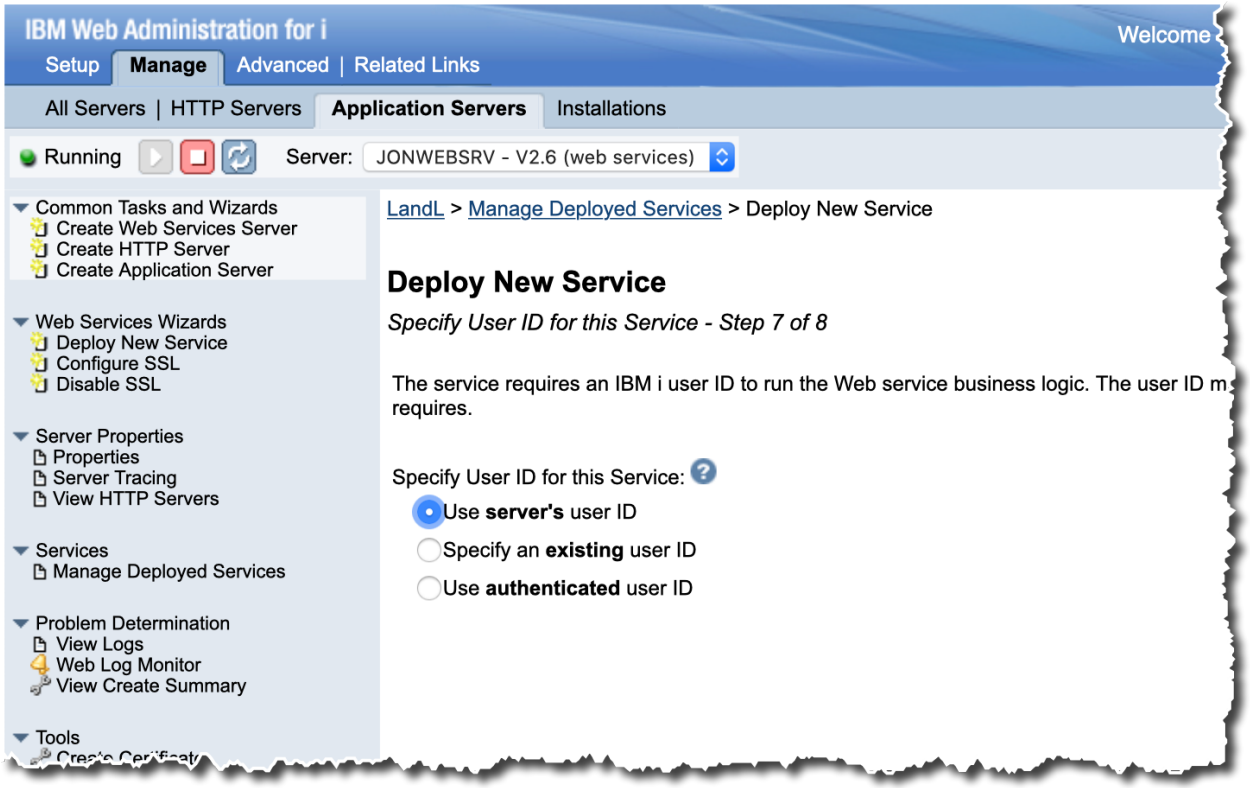
- The defaults for everything else are fine at this stage

Deploying an SQL Service (Cont)

Partner400

Last but not least specify the User Id to be used

- In most cases the default will be fine
- To use "authenticated" requires that you specified an HTTP server during the App Server configuration earlier



Deploying an SQL Service (Cont)

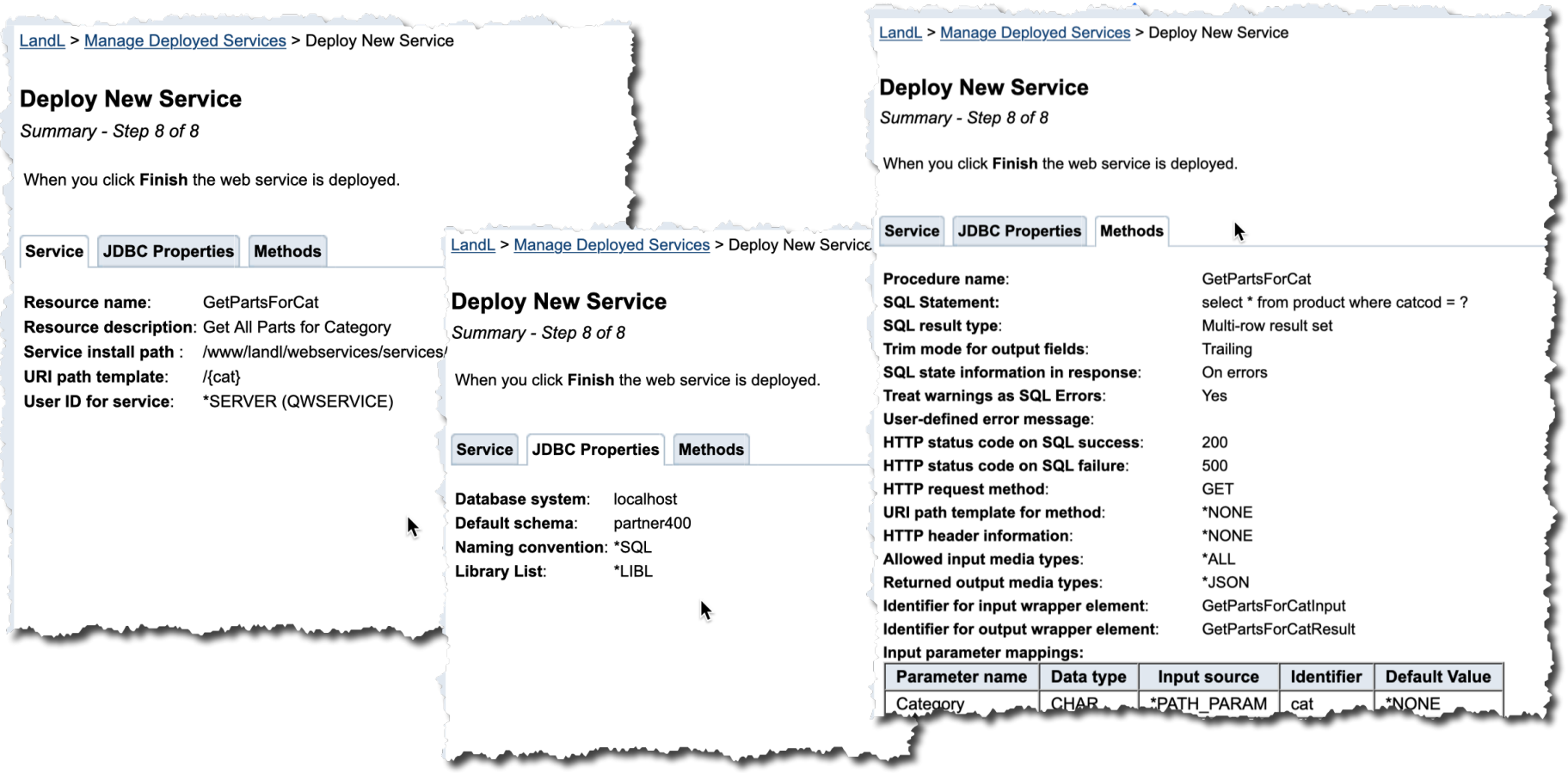
Partner400

This are the three tabs of the final confirmation screen

- You can use it to check that your details are correct

But in a simple case like this I will just press Finish

- That will install and launch the new service



Deploying an SQL Service (Cont)

Invoking the new REST service from the browser using

sidconf.idevcloud.com:10006/web/services/GetPartsForCat/02

Produces this result

- Note the name given to the resulting JSON array

Writing Your Own Web Service Code

My sample program is a very simple one

- It accepts a product category as an input parameter
- And returns the details of all the products in that category
  - Plus a message containing a count of the number of products

There are 2 versions

- The first returns a count of the number of entries plus the data
  - As you will see it may return blank entries depending on how the service is set up
- The second uses a count to limit the number of entries
  - It wo'nt return the actual count value

In both cases the parameter list is used to drive the interface

- The data is returned in a DS
  - This is the normal approach for this type of service

The next chart details the parameter list

- And addresses other essential settings in the program source

The RPG Program - Version 1

The actual logic I used is not terribly important

- If you care ... it is a simple SQL cursor loop
- The program parameters are the important thing

Plus you need these CTL-OPT (H-spec) options you probably don't normally use

- PGMINFO(*PCML : *MODULE)
  - This tells the compiler to store parameter details in the module
  - IWS will extract this information to build the interface
- DECEDIT('0.')
  - Prevents RPG from suppressing the leading zero before a decimal points
  - We need this because ' .5' is not a valid JSON number - it must be '0.5'

```
ctl-opt PGMINFO( *PCML : *MODULE ) DECEDIT('0.');

dcl-pi RestSrv4 ExtPgm;
   requestedCat         Like(catcod);
   result               char(20);
   productList_count    int(5);
   productList          LikeDS(product)  Dim(20);
end-pi;

Dcl-ds  product ExtName('PRODUCT')  End-ds;
```

Writing Your Own - Version 1 (Contd)

Parts Gartner400

First parameter is the requested category (input)

- Extracted from the URL by the IWS routines

The second is for an output text message from the program

- "No data found" or similar

The third is the count of the number of products returned

- Later versions move this into the list DS
  - You'll see why in a minute

Fourth one is the list of products

```
ctl-opt PGMINFO( *PCML : *MODULE ) DECEDIT('0.');

dcl-pi RestSrv4 ExtPgm;
   requestedCat         Like(catcod);
   result               char(20);
   productList_count    int(5);
   productList          LikeDS(product)  Dim(20);
end-pi;

Dcl-ds  product ExtName('PRODUCT')  End-ds;
```

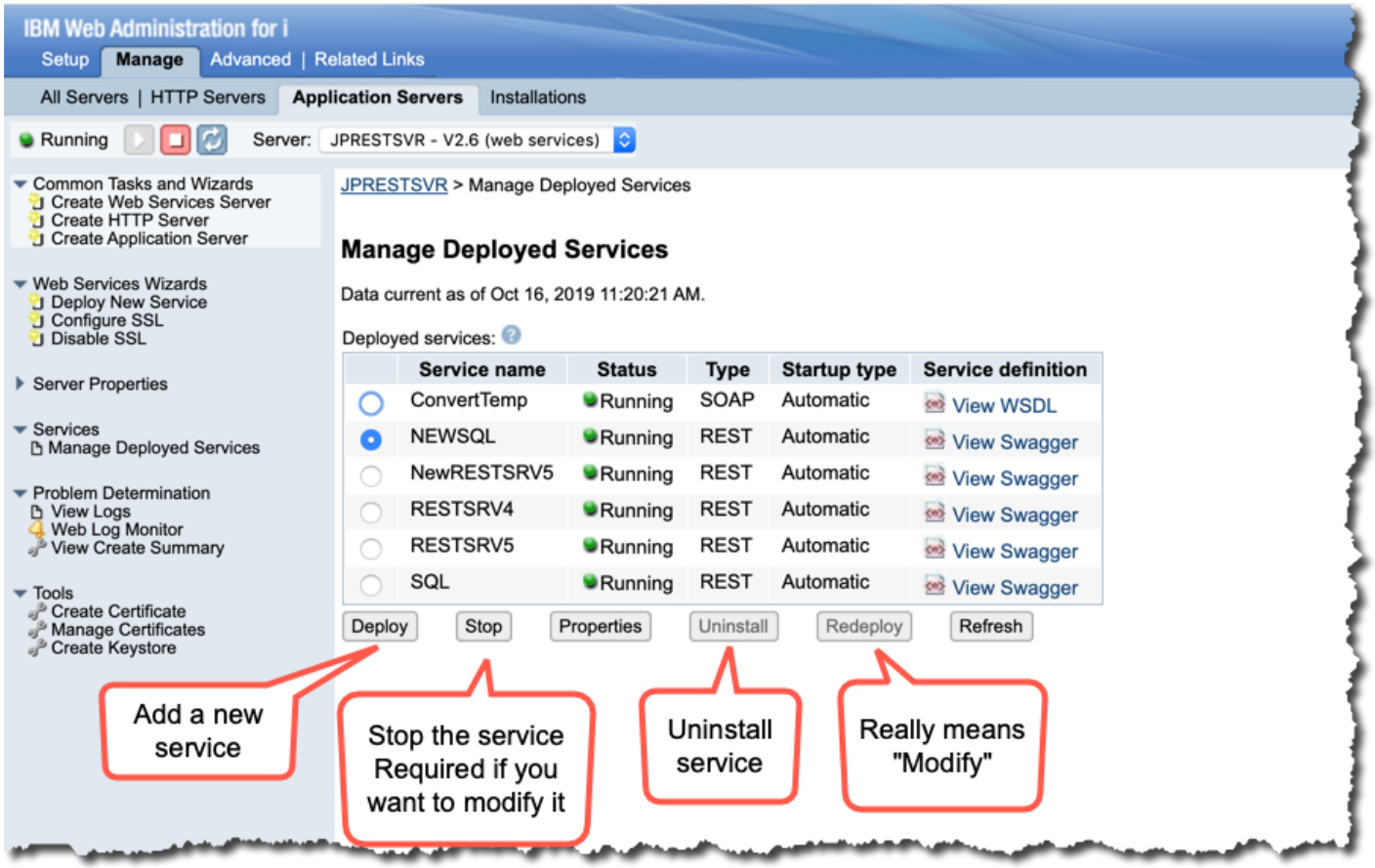
Note:
_count field is a
separate parameter

Service Management

Partner400

OK time to create a REST service from an RPG program

- We start the process by pressing "Deploy"



Deploying an RPG Service

Partner400

On the first page you select the type of service

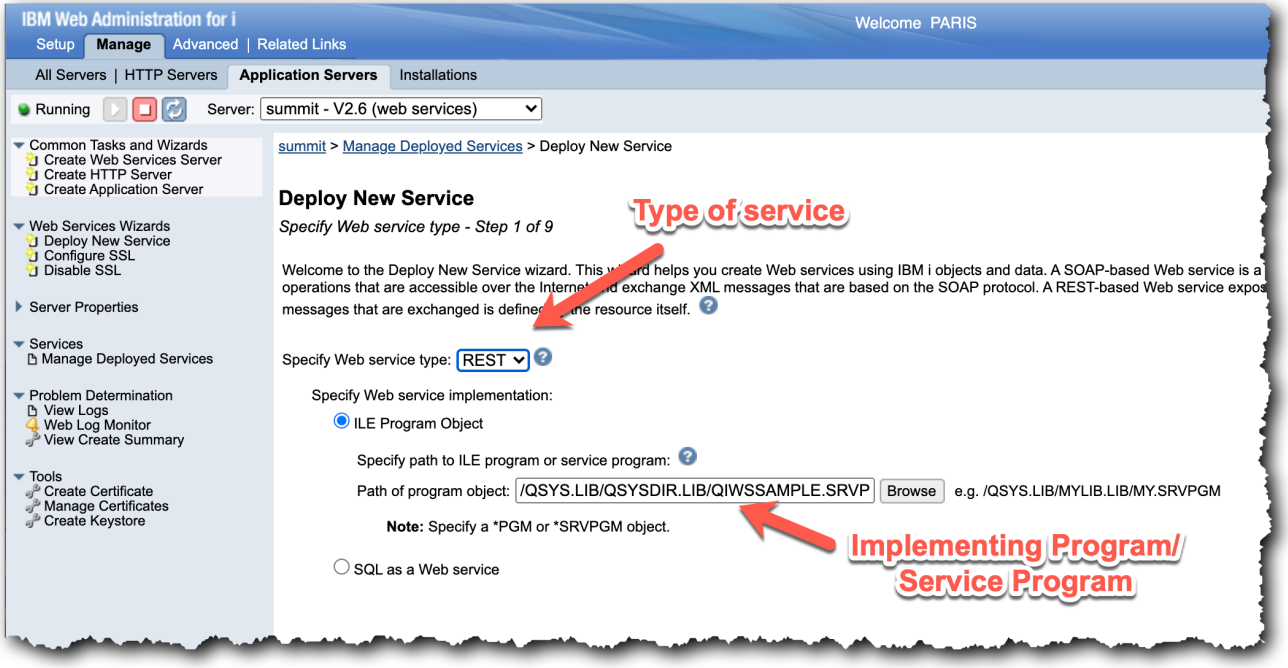
- REST (or if desired SOAP)

This time we specify implementation as being via a *PGM/*SRVPGM

- We also identify the path to the executable code

Note: I will not go through all the pages that work the same as in the SQL example

- I will just be highlighting the differences.



Deploying an RPG Service (Contd.)

Partner400

These were my initial parameter settings

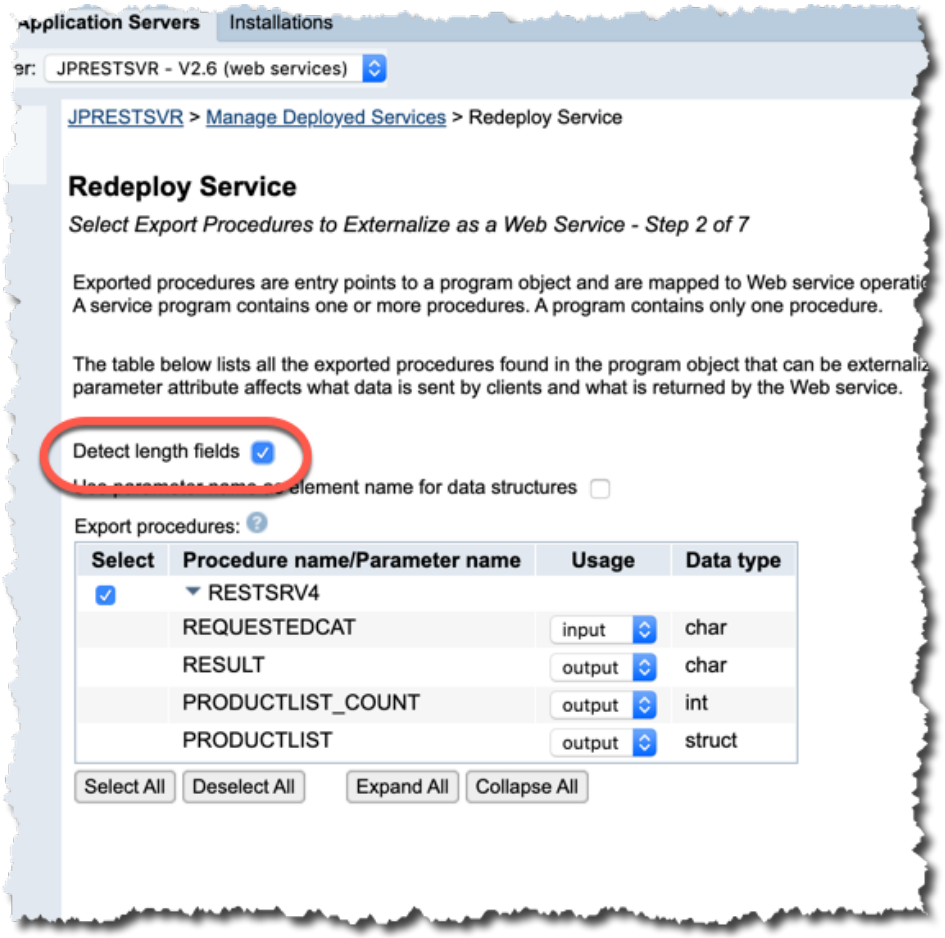
- Note "Detect Field Lengths"

But using this caused the entire array to be output

- Seems backwards doesn't it

This caused of all the empty product entries

We will see in a moment what happens when that is deselected ...



Deploying an RPG Service (Contd.)

Partner400

This is basically what the returned data looks like

- We'll see this live in a moment

The important thing is that my code did not build this JSON

- The code generated by the IWS wizard did

```
{
    "RESULT": "Found 1 rows for 01",
    "PRODUCTLIST_COUNT": 1,
    "PRODUCTLIST": [
        {
            "PRODCD": "0000020",
            "PRODDS": "A duplicate test",
            "CATCOD": "01",
            "STOH": 456,
            "LNDCST": 20.00,
            "SELLPR": 30.00,
            "DTLCHG": 107,
            "DTLORD": 990105
        },
        {
            "PRODCD": "",
            "PRODDS": "",
            ...
```

Data names are derived from the parameters

The _COUNT field is included in the output

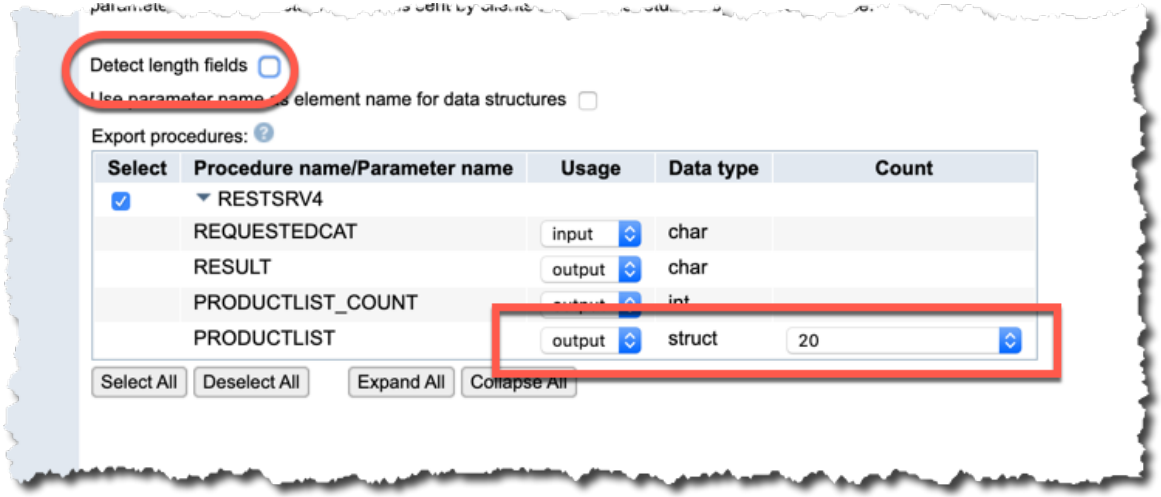
The data types are also derived from the parameters.
Note that the numeric fields are correctly represented

"Empty" entries.
This is not normally desirable or acceptable

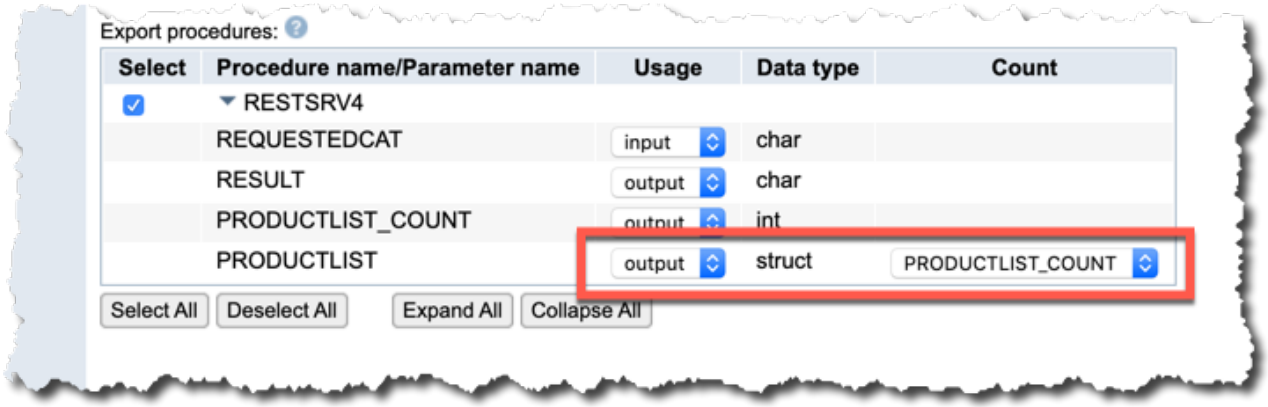
Deploying an RPG Service (Contd.)

Partner400

If I deselect the "Detect" box a length selection option appears



One of the values that can be selected is PRODUCTLIST_COUNT



Let's see the result of making that change

Deploying an RPG Service (Contd.)

Partner400

Notice the difference in the returned data

- No more empty entries
- Just the number specified by the count

In a moment we'll look at an alternative approach

- For cases where the count is not required as a value

```
{
    "RESULT": "Found 1 rows for 01",
    "PRODUCTLIST_COUNT": 1,
    "PRODUCTLIST": [
        {
            "PRODCD": "0000020",
            "PRODDS": "A duplicate test",
            "CATCOD": "01",
            "STOH": 456,
            "LNDCST": 20.00,
            "SELLPR": 30.00,
            "DTLCHG": 107,
            "DTLORD": 990105
        }
    ]
}
```

Array terminates after the last item

Deploying an RPG Service - Version 2

These are the parameter definitions for the second version

- Primary difference is in the positioning of the count
  - It is now WITHIN the result DS
- It must have the same base name as the array
  - The suffix _LENGTH tells the wizard to use it to control the returned length

Let's see how that affects the wizard and the subsequent results

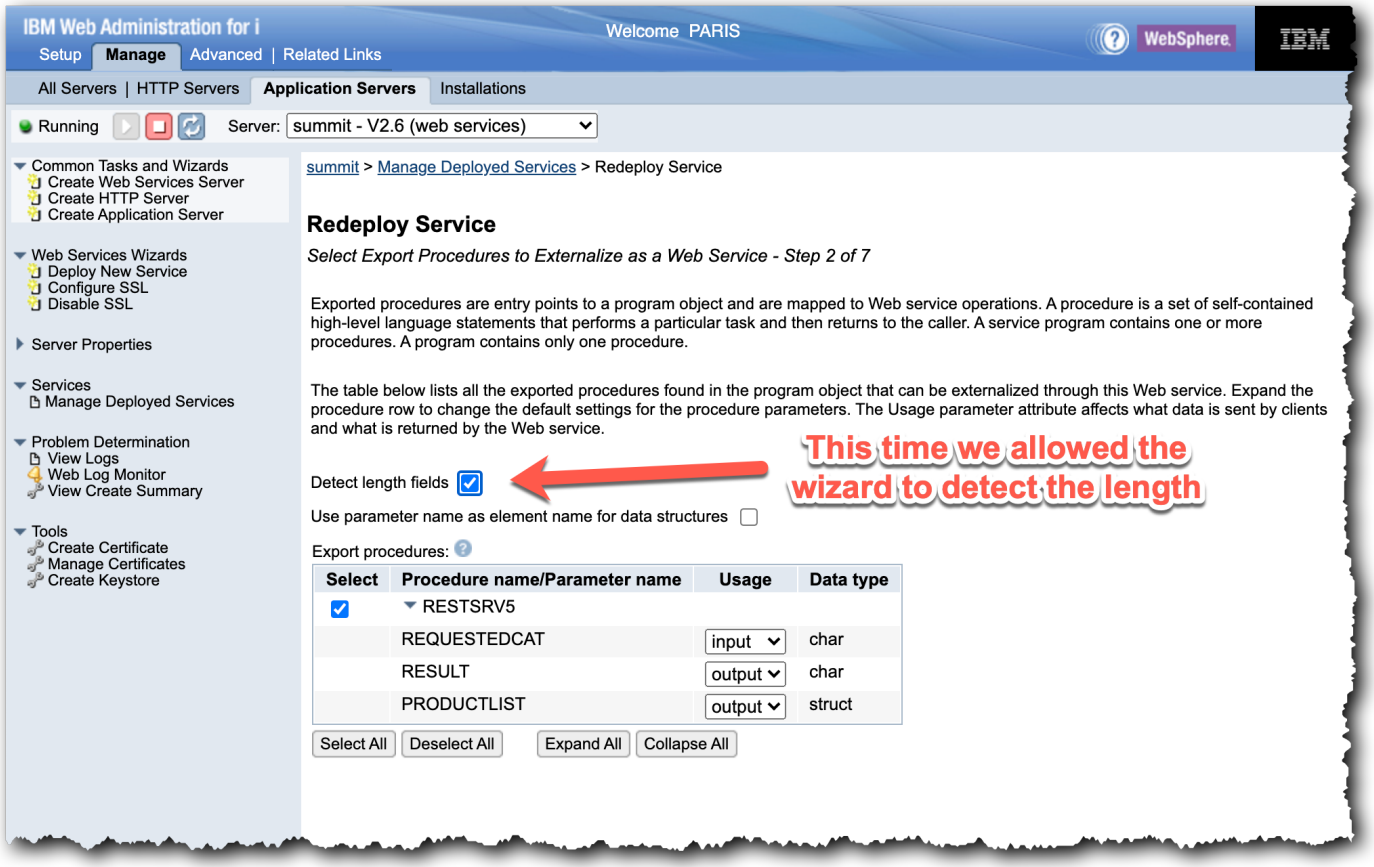
```
ctl-opt PGMINFO(*PCML : *MODULE) DECEDIT('0.'

dcl-pi RestSrv5 ExtPgm;
  requestedCat         Like(catcod);
  result               char(20);
  productList          LikeDS(productList_T);
end-pi;

Dcl-ds  product ExtName('PRODUCT')  End-ds;

Dcl-ds  productList_T  Qualified Template;
  products_LENGTH  int(5);
  products         LikeDS(product)  Dim(20);
End-Ds;
```

Deploying an RPG Service - Version 2 (Contd.)

Notice that this time we allow the wizard to detect the length

It will identify the field products_LENGTH

- And use it to control the number of products to output



Deploying an RPG Service (Contd.)

Partner400

Note the returned structure omits the count field

- But it has obviously been used to control the length of the output
  - No blank array entries!

```
{
    "RESULT": "Found 1 rows for 01",
    "PRODUCTLIST": {
        "PRODUCTS": [
            {
                "PRODCD": "0000020",
                "PRODDS": "A duplicate test",
                "CATCOD": "01",
                "STOH": 456,
                "LNDCST": 20.00,
                "SELLPR": 30.00,
                "DTLCHG": 107,
                "DTLORD": 990105
            }
        ]
    }
}
```

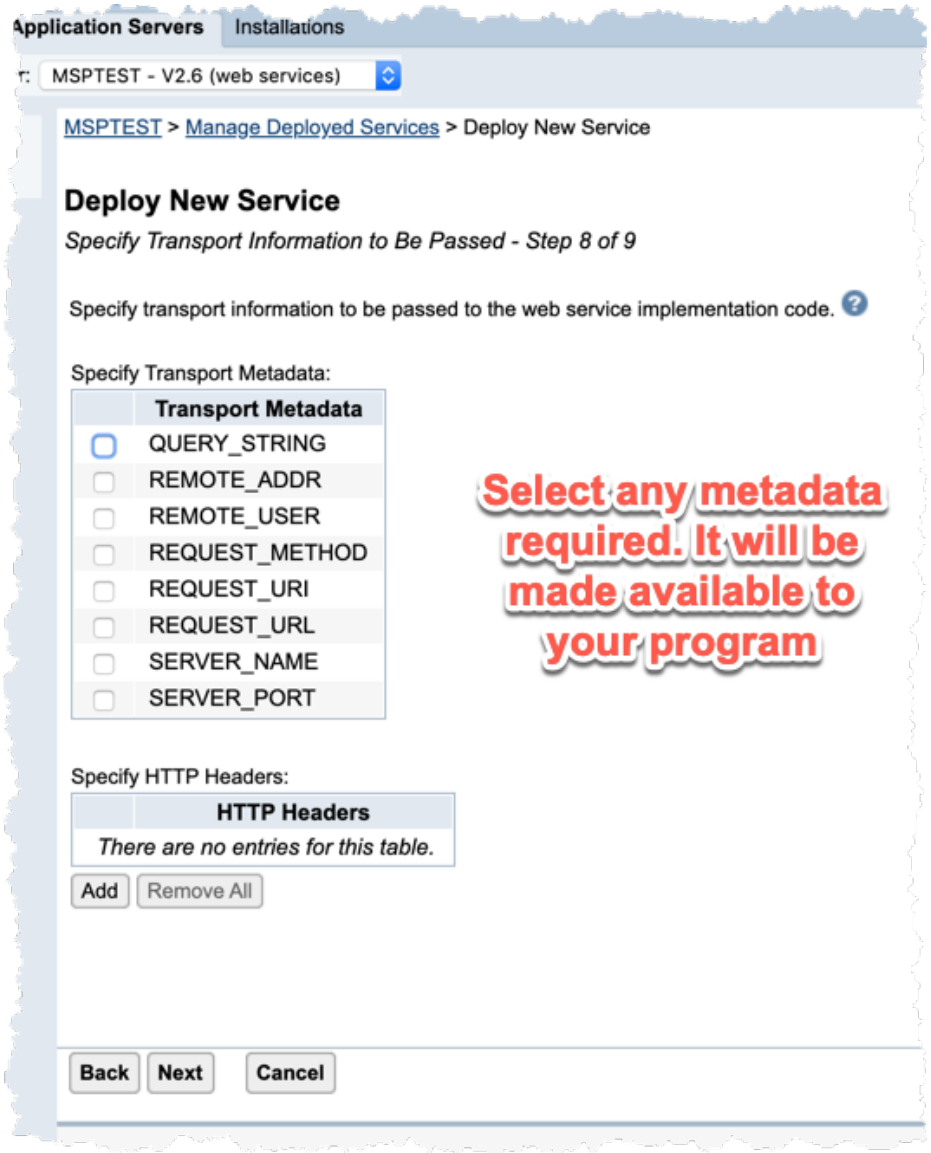
Additional Options for Deployed RPG Code

Partner400

More sophisticated services may need additional date from the request

- For example when the query string is used to supply data
- Or the remote user information is needed

Select items in this list to make them available to your RPG code



Last thoughts

Partner400

Qshell commands are now available for IWS services

- They can be used to to define and deploy, stop and start servers and services etc.
- They can control some aspects in a more granular fashion
  - For example the URL used

Links to the manuals for IWS can be found at:

- www.ibm.com/support/pages/node/633935

IWS also includes the ability to generate code to consume web services

- But personally I don't find it anywhere near as flexible as tools such as Scott Klement's HTTPAPI

Questions ????

Paris Partner400

email me Jon.Paris @ Partner400.com

Join us at the final Virtual RPG & Db2 Summit

systemideveloper.com/pages/events/RPGDb2Summit/